EECE-6820 Machine Learning

Gradient Descent

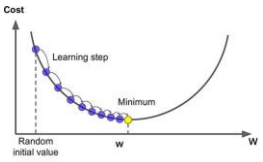
Cris Ababei
Dept. of Electrical and Computer Engineering
MARQUETTE UNIVERSITY
BE THE DIFFERENCE.

1

PART 1
A Lighter Presentation

2

What is Gradient Descent (GD)?



- When for (most) losses in practice there is no closed-form solution minimizer (such as the "Normal Equation"), we resort to numerical iterative methods.
- **Gradient Descent (GD):** Iterative optimization algorithm that gradually tweaks the model parameters to minimize the loss/cost function over the training set, eventually converging to the same set of parameters as the "Normal Equation".
- By repeatedly adjusting the model's parameters in the direction of the negative gradient (steepest descent), gradient descent helps the model learn and improve its accuracy.

3

3

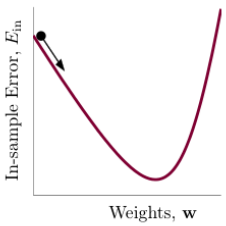
Iterative Optimization

For $t = 0, 1, \ldots$

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \mathbf{v}$$

when stop, return last $\mathbf{w}$ as g

- PLA: $\mathbf{v}$ comes from mistake correction
- smooth $E_{\text{in}}(\mathbf{w})$ for logistic regression: choose $\mathbf{v}$ to get the ball roll '**downhill**'?
  - direction $\mathbf{v}$: (assumed) of unit length
  - step size η: (assumed) positive



a greedy approach for some given $\eta > 0$:

$$\min_{\|\mathbf{v}\|=1} E_{\text{in}}(\underbrace{\mathbf{w}_t + \eta \mathbf{v}}_{\mathbf{w}_{t+1}})$$

4

4

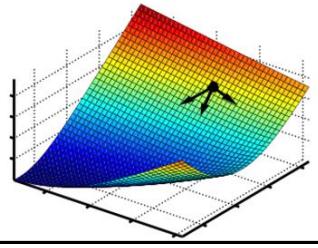
How to "Roll" Down?

Assume you are at weights $\mathbf{w}(t)$ and you take a step of size η in the direction $\hat{\mathbf{v}}$.

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta\hat{\mathbf{v}}$$

We get to pick $\hat{\mathbf{v}}$ ← what's the best direction to take the step?

Pick $\hat{\mathbf{v}}$ to make $E_{\text{in}}(\mathbf{w}(t+1))$ as small as possible.



5

Linear Approximation

a greedy approach for some given $\eta > 0$:

$$\min_{\|\mathbf{v}\|=1} \quad E_{\text{in}}(\mathbf{w}_t + \eta\mathbf{v})$$

- still non-linear optimization, now **with constraints**
  —not any easier than $\min_{\mathbf{w}} E_{\text{in}}(\mathbf{w})$
- local approximation by linear formula makes problem easier

$$E_{\text{in}}(\mathbf{w}_t + \eta\mathbf{v}) \approx E_{\text{in}}(\mathbf{w}_t) + \eta\mathbf{v}^T\nabla E_{\text{in}}(\mathbf{w}_t)$$

if η really small (Taylor expansion)

an **approximate** greedy approach for some given **small** η:

$$\min_{\|\mathbf{v}\|=1} \quad \underbrace{E_{\text{in}}(\mathbf{w}_t)}_{\text{known}} + \underbrace{\eta}_{\text{given positive}} \quad \underbrace{\mathbf{v}^T\nabla E_{\text{in}}(\mathbf{w}_t)}_{\text{known}}$$

6

6

The Gradient is the Fastest Way to Roll Down

an **approximate** greedy approach for some given **small** η:

$$\min_{\|\mathbf{v}\|=1} \underbrace{E_{\text{in}}(\mathbf{w}_t)}_{\text{known}} + \underbrace{\eta}_{\text{given positive}} \underbrace{\mathbf{v}^T \nabla E_{\text{in}}(\mathbf{w}_t)}_{\text{known}}$$

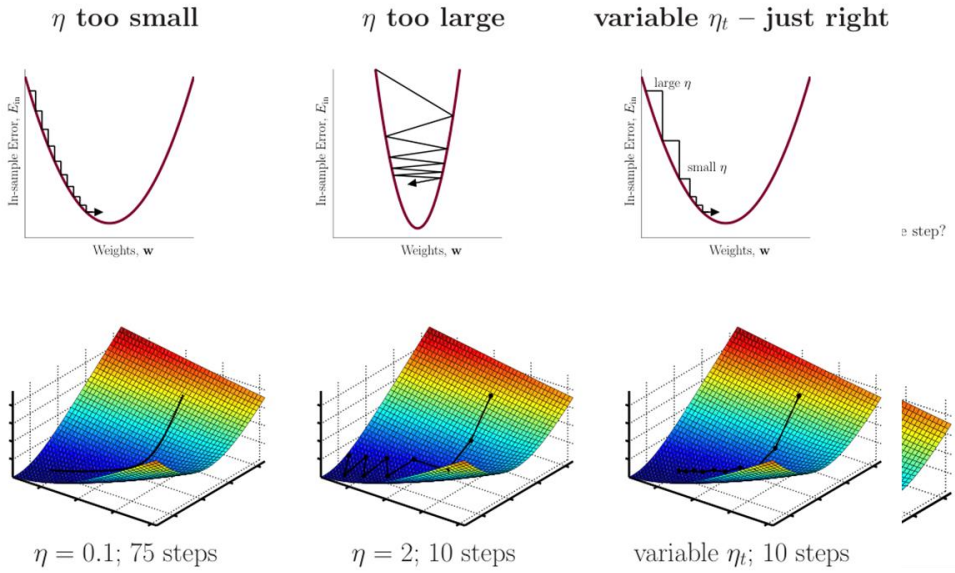
- optimal $\mathbf{v}$: opposite direction of $\nabla E_{\text{in}}(\mathbf{w}_t)$

$$\mathbf{v} = -\frac{\nabla E_{\text{in}}(\mathbf{w}_t)}{\|\nabla E_{\text{in}}(\mathbf{w}_t)\|}$$

- gradient descent: for **small** η, $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \frac{\nabla E_{\text{in}}(\mathbf{w}_t)}{\|\nabla E_{\text{in}}(\mathbf{w}_t)\|}$

gradient descent:
a simple & popular optimization tool

How to Choose Step Size



Fixed Learning Rate Gradient Descent (GD)

$$\eta_t = \eta \cdot \| \nabla E_{\text{in}}(\mathbf{w}(t)) \|$$

$\| \nabla E_{\text{in}}(\mathbf{w}(t)) \| \to 0$ when closer to the minimum.

$$\begin{aligned} \eta_t \hat{\mathbf{v}} &= -\eta_t \cdot \frac{\nabla E_{\text{in}}(\mathbf{w}(t))}{\| \nabla E_{\text{in}}(\mathbf{w}(t)) \|} \\ &= -\eta \cdot \cancel{\| \nabla E_{\text{in}}(\mathbf{w}(t)) \|} \cdot \frac{\nabla E_{\text{in}}(\mathbf{w}(t))}{\cancel{\| \nabla E_{\text{in}}(\mathbf{w}(t)) \|}} \end{aligned}$$

$$\eta_t \hat{\mathbf{v}} = -\eta \cdot \nabla E_{\text{in}}(\mathbf{w}(t))$$

1: Initialize at step $t = 0$ to $\mathbf{w}(0)$.
2: **for** $t = 0, 1, 2, \ldots$ **do**
3: Compute the gradient
$$\mathbf{g}_t = \nabla E_{\text{in}}(\mathbf{w}(t)).$$ ← (Ex. 3.7 in LFD)
4: Move in the direction $\mathbf{v}_t = -\mathbf{g}_t$.
5: Update the weights:
$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \mathbf{v}_t.$$
6: Iterate 'until it is time to stop'.
7: **end for**
8: Return the final weights.

Gradient descent can minimize any smooth function, for example

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \ln(1 + e^{-y_n \cdot \mathbf{w}^{\mathrm{T}} \mathbf{x}})$$ ← logistic regression

Stochastic Gradient Descent (SGD)

A variation of GD that considers only the error on one data point.

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \ln(1 + e^{-y_n \cdot \mathbf{w}^{\mathrm{T}} \mathbf{x}}) = \frac{1}{N} \sum_{n=1}^{N} e(\mathbf{w}, \mathbf{x}_n, y_n)$$

- Pick a random data point $(\mathbf{x}_*, y_*)$
- Run an iteration of GD on $e(\mathbf{w}, \mathbf{x}_*, y_*)$

$$\mathbf{w}(t+1) \leftarrow \mathbf{w}(t) - \eta \nabla_{\mathbf{w}} e(\mathbf{w}, \mathbf{x}_*, y_*)$$

1. The 'average' move is the same as GD;
2. Computation: fraction $\frac{1}{N}$ cheaper per step;
3. Stochastic: helps escape local minima;
4. Simple;
5. Similar to PLA.

Logistic Regression:

$$\mathbf{w}(t+1) \leftarrow \mathbf{w}(t) + y_* \mathbf{x}_* \left(\frac{\eta}{1 + e^{y_* \mathbf{w}^{\mathrm{T}} \mathbf{x}_*}} \right)$$

(Recall PLA: $\mathbf{w}(t+1) \leftarrow \mathbf{w}(t) + y_* \mathbf{x}_*$)

Stochastic Gradient Descent (SGD)

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \underbrace{\frac{1}{N}\sum_{n=1}^{N} \theta\left(-y_n \mathbf{w}_t^T \mathbf{x}_n\right)\left(y_n \mathbf{x}_n\right)}_{-\nabla E_{\text{in}}(\mathbf{w}_t)}$$

- want: update direction $\mathbf{v} \approx -\nabla E_{\text{in}}(\mathbf{w}_t)$
  while computing $\mathbf{v}$ by one single $(\mathbf{x}_n, y_n)$
- technique on removing $\frac{1}{N}\sum_{n=1}^{N}$:
  view as expectation $\mathcal{E}$ over uniform choice of n!

stochastic gradient:
$\nabla_{\mathbf{w}} \ \text{err}(\mathbf{w}, \mathbf{x}_n, y_n)$ with random n
true gradient:
$\nabla_{\mathbf{w}} \ E_{\text{in}}(\mathbf{w}) = \underset{\text{random } n}{\mathcal{E}} \ \nabla_{\mathbf{w}} \ \text{err}(\mathbf{w}, \mathbf{x}_n, y_n)$

Stochastic Gradient Descent (SGD)

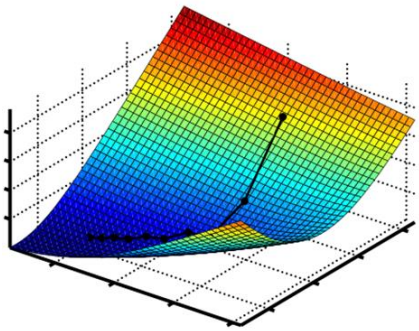
stochastic gradient = true gradient + zero-mean 'noise' directions

Stochastic Gradient Descent

- idea: replace true gradient by stochastic gradient
- after enough steps,
  average true gradient ≈ average stochastic gradient
- pros: **simple & cheaper computation :-)**
  —useful for **big data** or **online learning**
- cons: less stable in nature

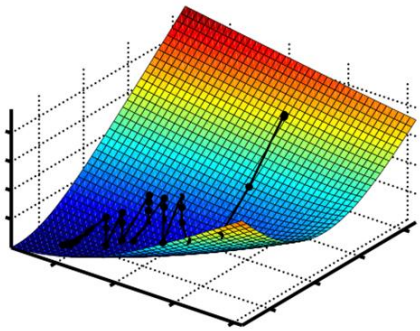
Stochastic Gradient Descent (SGD)

GD



$\eta = 6$
10 steps
$N = 10$

SGD



$\eta = 2$
30 steps

PART 2
More Details

Details

- See hand-written notes and discussion in class.

15

PART 3
Code Time!

16

Code Time

- See assignment and discussion in class.

17

17

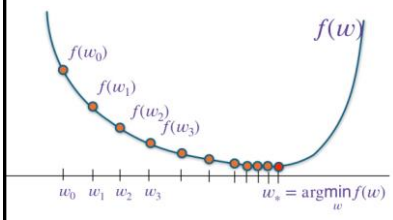
Conclusion
Takeaways

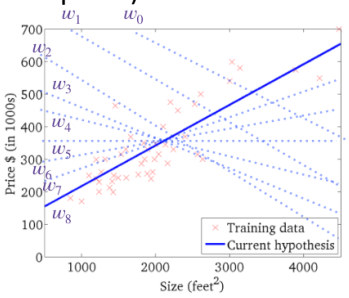
18

18

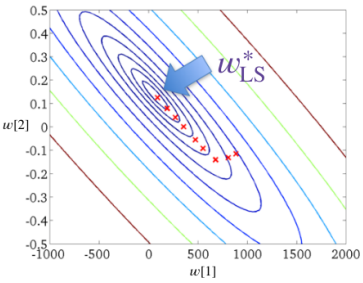
Conclusion

- Main idea of GD:
  - Start with an initial guess for model's parameters (weights).
  - Compute gradient (vector of partial derivatives) of the loss with respect to the parameters. The gradient points in the direction of steepest increase.
  - Update the parameters by moving them in the opposite direction of the gradient, scaled by a learning rate (a small step size).





Evolution of the predictor y = w[0] + w[1]x



Gradient descent dynamics in the parameter space w
Ovals show the **level set** of the objective function

19

References and Credits

Many of the teaching materials for this course have been adapted from various sources. We are very grateful and thank the following professors, researchers, and practitioners for sharing their teaching materials (in no particular order):

- Yaser S. Abu-Mostafa, Malik Magdon-Ismail and Hsuan-Tien Lin. https://amlbook.com/slides.html
- Ethem Alpaydin. https://www.cmpe.boun.edu.tr/~ethem/i2ml3e/
- Natasha Jaques. https://courses.cs.washington.edu/courses/cse446/25sp/
- Lyle Ungar. https://alliance.seas.upenn.edu/~cis520/dynamic/2022/wiki/index.php?n=Lectures.Lectures
- Aurelien Geron. https://github.com/ageron/handson-ml3
- Sebastian Raschka. https://github.com/rasbt/machine-learning-book
- Trevor Hastie. https://www.statlearning.com/resources-python
- Andrew Ng. https://www.youtube.com/playlist?list=PLoROMvodv4rMiGQp3WXShtMGgzqpfVfbU
- Richard Povineli. https://www.richard.povinelli.org/teaching
- … and many others.

20

20